Influxis Apps Skinning/Customizing v1.1

Updated: 2/20/2013

Overview

The latest Influxis apps have a very flexible API which allows them to be skinned or customized in a variety of ways. The API skinning works from an XML file that is loaded into the application (skins.xml). The application API will load the new skin file at run time so no new code needs to be compiled to carry out changes. The XML structure is composed of **skin** tags. The skin tags have nested style and **styleItem** tags to change certain aspects of the app's components.

```
<skins>
	<skin instanceID="org.influxis.application.InfluxisApplication">
		<style styleName="background">
			<styleItem styleName="backgroundColor">black</styleItem>
			<styleItem styleName="borderSides"></styleItem>
		</style>
	</skin>
</skins>
```

The skin tag targets a display component within the app. The style tag targets a display item within the target component. The styleItem tag targets certain properties to the target style. So for example the XML structure above points to the whole InfluxisApplication component. Its style tag is targeting the background of the component. The styleItem tags are then setting the background to black with no border appearing. Not all style tags target display graphics so the styleItem tag may not be available in those cases.

There are three main styles used in the API:

1. The first is for drawing graphics. This could be backgrounds, buttons, or icons.
2. The second is for text fields which hold labels.
3. The third is for everything else and this can be anything such as numerical styles for determining gaps or width/height sizes.

Working with Graphics

There are three ways to set up graphics:

1. The first way is by pointing to a symbol within a SWF file. This can be done by enclosing the file path, followed by a colon, and then the symbol name in the style tag. Our example above would look like this:

   ```
   <skin instanceID=" org.influxis.application.InfluxisApplication ">
       <style styleName="background">osmfskins.swf:backgroundSymbol</style>
   </skin>
   ```

2. The second way is by pointing the path to a jpg, png, swf, or gif file. The above example would look like this:

   ```
   <skin instanceID=" org.influxis.application.InfluxisApplication ">
       <style styleName="background">playerBackground.jpg</style>
   </skin>
   ```

3. The last way to handle graphics is by using the styleItem tags to specify which properties the graphic should have. Refer to the first example above. The supported styleNames for the styleItem tags are:

   borderSides:
   Tells the API which border sides to show.
   Value: left right top bottom

   borderThickness:
   Tells the API how thick the border should be.
   Value: Any numerical value

   borderColor:
   Tells the API which border color to use.
   Value: Any hex numerical value (0x0 or #0) as well as general color names (black, blue, etc). Multiple colors can be entered separated by comma to create a gradient.

   borderAlpha:
   Tells the API which alpha to apply to the border.
   Value: Numerical value from 0 to 1. If multiple colors are being used then multiple alphas separated by comma can also be applied for each color.

   borderRatios:
   If using gradients, tells the API how to spread the gradient across the border.
   Value: Numerical value from 0 to 255. If gradient is being used multiple values can be inserted for each color separated by color.

   backgroundColor:
   Tells the API which background color to apply.
   Value: See borderColor.

   backgroundAlpha:
   Tells the API which alpha to apply to the background.
   Value: See borderAlpha.

   backgroundRatios:
   If using gradients, tells the API how to spread the gradient across the background.
   Value: See borderRatios.

   cornerRadius:
   Tells the API how much corner radius should be applied to the rectangle graphic for rounded corners.
   Value: Any numerical value.

   gradientType:
   Tells the API which gradient type to apply (if using gradients)
   Value: linear or radial.

   angle:
   Tells the API what angle the gradient should be drawn at.

Value: Numerical value 0 to 360.

image:
Tells the API to add an image to the background.
Value: Same as adding an image to the main style tag. Accepts swf, swf graphics, png, gif, and jpg.

mask:
Tells the API to add a mask image to the foreground.
Value: Same as adding an image to the main style tag. Accepts swf, swf graphics, png, gif, and jpg

Filters

The API also supports three filter effects that can be used with **styleItem** graphics and text fields:

Shadow:

dropShadowEnabled:
Tells the API if it should apply a shadow effect to the target graphic.
Value: true or false.

shadowAlpha:
Tells the API to set the shadow alpha channel to make the shadow more transparent or opaque.
Value: A floating numerical value from 0 to 1.

shadowAngle:
Tells the API at what angle to render the shadow.
Value: A numerical value from 0 to 360.

shadowBlurX:
Tells the API how much blur effect along the x axis to apply to the shadow.
Value: Any numerical.

shadowBlurY:
Tells the API how much blur effect along the y axis to apply to the shadow.
Value: Any numerical value.

shadowColor:
Tells the API what color to apply to the shadow.
Value: See borderColor (note: Multiple colors not supported).

shadowDistance:
Tells the API how far to spread the shadow effect.
Value: Any numerical value.

shadowInner:
Tells the API whether to spread the shadow inside or outside.
Value: true or false.

shadowStrength:
Tells the API how much strength to apply to the shadow.
Value: Any numerical value.

shadowQuality:
Tells the API what quality it should apply to the shadow.
Value: Any numerical value.

Glow:

glowEnabled:
Tells the API if it should apply a glow effect to the target graphic.
Value: true or false.

glowAlpha:
Tells the API to set the glow alpha channel to make the glow more transparent or opaque.
Value: A floating numerical value from 0 to 1.

glowBlurX:
Tells the API how much blur effect along the x axis to apply to the glow.
Value: Any numerical.

glowBlurY:
Tells the API how much blur effect along the y axis to apply to the glow.
Value: Any numerical value.

glowColor:
Tells the API what color to apply to the glow.
Value: See borderColor (note: Multiple colors not supported).

glowInner:
Tells the API whether to spread the shadow inside or outside.
Value: true or false.

glowStrength:
Tells the API how much strength to apply to the glow.
Value: Any numerical value.

glowQuality:
Tells the API what quality it should apply to the glow.
Value: Any numerical value.

Blur:

blurEnabled:
Tells the API if it should apply a blur effect to the target graphic.
Value: true or false.

blurBlurX:

Tells the API how much blur effect along the x axis to apply to the blur.
Value: Any numerical value.

blurBlurY:
Tells the API how much blur effect along the y axis to apply to the blur.
Value: Any numerical value.

blurQuality:
Tells the API what quality it should apply to the blur.
Value: Any numerical value.

Setup Text Field Styles

There are two ways to setup label text fields in the API:

1. The first way is by pointing to a symbol within a SWF file just like the graphic example. This can be done by enclosing the file path, followed by a colon, and then the symbol name in the style tag. The symbol in the compiled SWF must contain a text field with all the properties for the text already set on it. Furthermore there has to be some dummy fake text set for the system to catch it.

   Here is an example:

   ```
   <skin instanceID="org.influxis.as3.display.SimpleMediaControls">
           <style styleName="timeLabel">osmfskins.swf:SimpleMediaControlsSkin_timeLabel</style>
   </skin>
   ```

The system will catch all the formats applied to the text and will apply them to the target textfield style.

2. The second way to set text properties is by using the **styleItem** tags. The following **styleNames** are supported:

   - align – Any numerical value
   - blockIndent – Any numerical
   - bold – true or false
   - color – Any hex value
   - font – Font name
   - indent – Any numerical value
   - italic – true or false
   - kerning – Any numerical value
   - leading – Any numerical value
   - leftMargin – Any numerical value
   - letterSpacing – Any numerical value
   - rightMargin – Any numerical value
   - size – Any numerical value
   - underline – true or false
   - sharpness - Any numerical value
   - thickness - Any numerical value
   - antiAliasType – advanced or normal

Here is an example of the label properties being used:

```
<skin instanceID="org.influxis.as3.display.SimpleMediaControls">
	<style styleName="timeLabel">
		<styleItem styleName=”color”>blue<styleItem>
		<styleItem styleName=”size”>12<styleItem>
	</style>
</skin>
```

Also when using **styleItem** tags, the same filters applied to graphics can also be applied to text fields. See graphic filters for more info.

Other Styles

The API also supports styles to change or customize certain aspects of the interface or functionality. Make sure to check out the skins.xml for more details.

Applying Changes

Once all the changes have been made, save the XML file. Run it in a browser to make sure there are no errors in the structure. Include the new XML with the main player SWF or where the SWF is embedded. Make sure to include any assets included in the XML API. If the new XML file is not called “skins.xml”, then make sure to use the “skin” setting to indicate where the new skin XML file is located. Reload the player and if everything was done correctly, the player should catch the changes.